

Index Optimization with Combination of Text Categorization by Table based KNN

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the table based KNN version for implementing the fully automatic index optimization system. In the previous work, although the table based KNN version was proposed as an approach to the index optimization, the domain should be presented manually as a tag, together with the input text. In the proposed system, only a text is given without its domain tag, the domain is automatically assigned to the text, through the text classification, and the words are classified in the text by the classifier which corresponds to the domain. The table based KNN algorithm which receives a table instead of a numerical vector is adopted as the approach to the text classification and the index optimization. The goal of this research is to automate the index optimization without presenting a domain tag and improve the performance of both tasks, using the table based KNN algorithm.

I. INTRODUCTION

The index optimization is the process of optimizing the text indexes by the index expansion and the index filtering. In the task, it is necessary to define words into the three classes, and the task is mapped into the classification task where each word is classified into one of expansion, inclusion, and removal. The index expansion which is the process of adding associative words from external sources are applied to the words which are classified into expansion, and the words which are classified into removal are excluded from the indexes as the index filtering. The classification task which is mapped from the index optimization belongs to the domain dependent classification where an entity may be classified differently depending on the domain. The domain should be presented as a tag for nominating a classifier which corresponds to the domain.

This research is motivated by the requirement of presenting the domain for executing the index optimization. It is defined as an independent classification task within each domain, and a classifier is allocated to each domain, independently of others. It is necessary to know the domain in advance for selecting a classifier which corresponds to it. Only text is provided for doing the keyword extraction by automating the process of assigning a domain to a text through the text classification. The solution to the requirement is the connection of the keyword extraction with the text classification.

The table based KNN algorithm is adopted for implementing the index optimization by connecting it with the

text classification. We define the similarity metric between tables and modify the KNN algorithm into the version which processes tables directly, based on the similarity metric. The modified KNN algorithm is applied to the text classification which assigns a domain to an input text automatically. A text is indexed into words, and the modified KNN algorithm is applied to the classification of words into one of the three categories. This research is intended to free from labeling manually an input text with its own domain for nominating a classifier.

Let us mention some benefits from this research. The problems in encoding words or texts into numerical vectors are solved by encoding them into alternative structured forms, tables. The performance in both the index optimization and the text classification is improved by adopting the table based KNN algorithm. It does not require to present the domain manually for executing the index optimization. We automate the process of deciding a suitable classifier as upgrading the index optimization system.

Let us mention remaining tasks for reinforcing this research. We may define more operations on tables as well as the similarity metric between them. Words and texts may be represented into compound forms, each of which consists of more than one structured form. Other machine learning algorithms and deep learning algorithms may be modified into table-based versions. The index optimization system may be implemented into a real program or a library by adopting the table based KNN algorithm.

II. PROPOSED APPROACH

A. Similarity Metric

This section is concerned with the table as the representation of a word. The table is defined as a set of entries, each of which consists of an element and its weight. The table which represents a word consists of entries each of which contains a text identifier and its weight. The similarity between tables is computed based on shared words as the semantic similarity between words. This section is intended to describe the process of encoding a word into a table, and the process of computing the similarity between tables.

The process of encoding words into tables is illustrated in Figure 1. In the inverted indexing, each word is linked with texts which include itself. In the text-word weighting, its

weights in the lined texts are computed; a weight indicates a relationship between a word and a text. Entries with their lower weights are removed for downsizing the table. Refer to [1] for studying the encoding process in more detail.

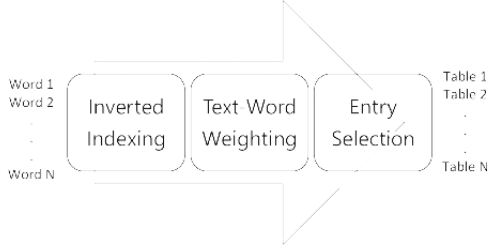


Figure 1. Process of Encoding Words into Tables

Let us mention the function of a table which maps it into a set of text identifiers as shown in Figure 2. The table which represents a word is expressed as entry set as shown in equation (1),

$$T = \{(d_1, w_1), (d_2, w_2), \dots, (d_{|T|}, w_{|T|})\} \quad (1)$$

where d_i is a text identifier and w_i is the weight of the word, T , in the text, d_i . The function for generating a list of text identifiers is expressed as equation (2),

$$F(T) = \{d_1, d_2, \dots, d_{|T|}\} \quad (2)$$

The function is the operation which takes only text identifiers without their weights as a set. The operation is applied to computing the similarity between tables which represent words.

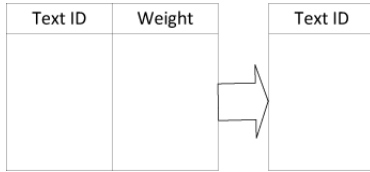


Figure 2. Conversion of Table into Text Set

Let us mention the process of computing the similarity between two tables as a semantic similarity between words. The two tables which represent words are notated respectively by $T_1 = \{(d_{11}, w_{11}), (d_{12}, w_{12}), \dots, (d_{1|T_1|}, w_{1|T_1|})\}$ and $T_2 = \{(d_{21}, w_{21}), (d_{22}, w_{22}), \dots, (d_{2|T_2|}, w_{2|T_2|})\}$. The intersection between the two sets which is generated by applying the function, $F(\cdot)$ as shown in equation (3),

$$F(T_1) \cap F(T_2) = \{d_{s1}, d_{s2}, \dots, d_{s|F(T_1) \cap F(T_2)|}\} \quad (3)$$

and the table with entries each of which consists of a shared text identifier, d_{si} , its weight from the table, T_1 , w_{si}^1 , and its weight from the table, T_2 , w_{si}^2 , is expressed as equation

(4),

$$ST_{12} = \{(d_{s1}, w_{s1}^1, w_{s1}^2), (d_{s2}, w_{s2}^1, w_{s2}^2), \dots, (d_{s|F(T_1) \cap F(T_2)|}, w_{s|F(T_1) \cap F(T_2)|}^1, w_{s|F(T_1) \cap F(T_2)|}^2)\} \quad (4)$$

The similarity between tables, T_1 and T_2 is computed by equation (5),

$$sim(T_1, T_2) = \frac{2(\sum_{i=1}^{|F(T_1) \cap F(T_2)|} w_{si}^1 + \sum_{i=1}^{|F(T_1) \cap F(T_2)|} w_{si}^2)}{\sum_{i=1}^{|T_1|} w_{1i} + \sum_{i=1}^{|T_2|} w_{2i}}. \quad (5)$$

The value which is computed from equation (5) is always given as a normalized value between zero and one as shown in equation (6),

$$0 \leq sim(T_1, T_2) \leq 1. \quad (6)$$

Let us make some remarks on the encoding process and the computation of similarity between words. A word is encoded into table by the inverted indexing, the text identifier weighting, and the entry selection. A table is expressed as a set of entries and filtered into a set of text identifiers by the function. The similarity between tables is computed based on their shared entries by equation (5). In the future research, we consider representing a word into multiple tables.

B. Table based KNN Algorithm

This section is concerned with table based KNN algorithm. It was initially proposed as the approach to the word classification by Jo in 2018 [1]. The similarity metric between tables which was described in the previous section is used for computing the similarity between a novice table and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the table-based version of KNN algorithm.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 3. The labeled words which are gathered as samples are encoded into tables, and they are notated by a set $Tr = \{T_1, T_2, \dots, T_N\}$. A novice word is encoded into a table notated by T . Its similarities with the tables which represent the sample words are computed by equation (5), as $sim(T, T_1), sim(T, T_2), \dots, sim(T, T_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice

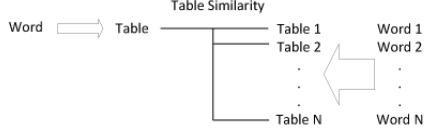


Figure 3. Similarities of Novice Input with Training Examples

example are selected as its nearest neighbors, as expressed in equation (7),

$$Nr = \text{Select}_k(Tr, T), Nr \subseteq Tr \quad (7)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (8),

$$Nr = \{T_1^{near}, T_2^{near}, \dots, T_k^{near}\} \quad (8)$$

The elements in the set, Nr , are used for voting their labels in the next step.

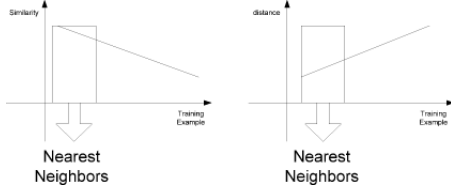


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(T_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, T , is decided by equation (9),

$$\text{label}(T) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (9)$$

The process of deciding the label of a novice item by equation (9), is called voting.

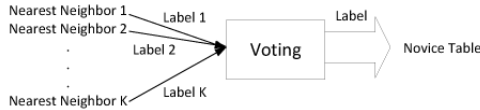


Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the table-based version of KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the index optimization system which adopts the table based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the index optimization. It is viewed as a classification task which classifies a word into one of the three categories. This section is intended to describe the index optimization system with respect to its function and its architecture.

The process of collecting the same words and encoding them into tables for implementing the index optimization system is illustrated in Figure 6. The index optimization is interpreted into a domain dependent classification; even a same word is classified differently, depending on the domain. For each domain, an independent classification task is defined. The several domains are decided, and the words which are labeled with one of the three categories exclusively in each domain. It is required to present the domain from a text which is given as the input for doing this task.

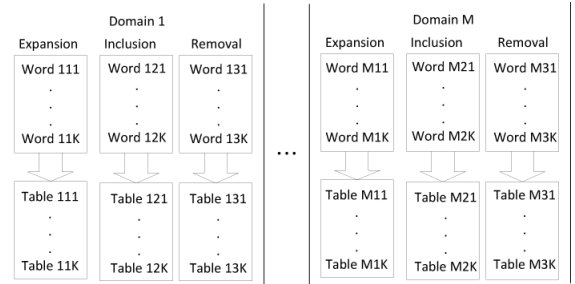


Figure 6. Preparation of Training Examples

The entire architecture of the proposed index optimization system is illustrated in Figure 7. A text is given as the input, and words are extracted from it in the indexing module. In the encoding module, the sample words in the groups, expansion, inclusion, and removal and ones which are indexed from the text are encoded into tables. In the similarity computation module and the voting module, the words which are indexed from the text are classified into one of the three categories. The words which are classified into removal are discarded in this system.

The execution process of the proposed system is illustrated as a block diagram in Figure 8. Sample words which are labeled with one of the three categories are collected within a domain, and they are encoded into tables. An input text is indexed into a list of words, and they are also encoded into tables. The nearest neighbors are selected by the similarity computation, and the label is decided by voting ones of their nearest neighbors for each word. Ones which are classified into expansion require their associated words from external sources, ones which are classified into inclusion are preserved, and ones which are classified into

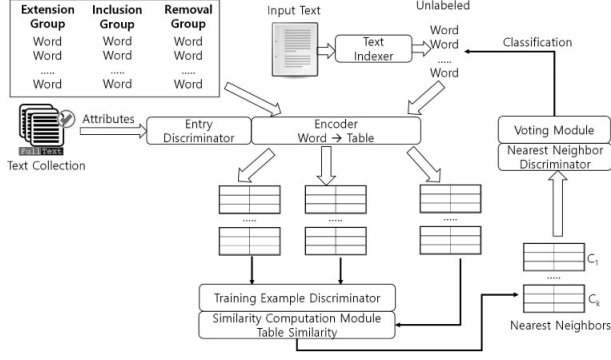


Figure 7. System Architecture

removal are excluded.

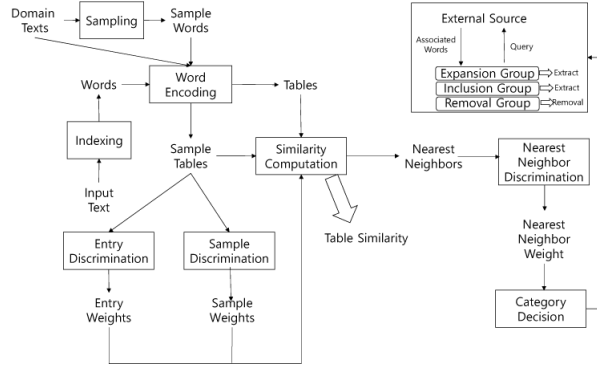


Figure 8. System Flow

Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The index optimization is defined as a multiple classification task where each word is classified into expansion, inclusion, or removal. The words are encoded into tables instead of numerical vectors, and they are classified directly. The words which are classified into removal are excluded from index of the input text. In implementing the system as its complete version, we need to add the module which retrieve more words which are relevant to ones which are labeled with expansion.

D. Connection with Text Classification

This section is concerned with the connection of the index optimization with the text classification. In the previous section, we mentioned the index optimization system as an instance of domain dependent classification. The domain is automatically decided for nominating a classifier by connecting the keyword with the text classification. The KNN algorithm which was described in Section II-B is used for implementing the text classification. This section is intended to describe the connection of the index optimization system with the text classification for automating the decision of the domain.

The system architecture of the text categorization system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 9. The encoding module is for encoding texts into tables by the process which is described in Section II-A. The similarity computation module is for computing the similarities between tables which represent a novice text and a sample text and selecting ones with their highest similarities as the nearest neighbors. The voting module is for deciding the label of the novice item by voting the labels of the nearest neighbors. This system used for automating deciding the domain of the input text.

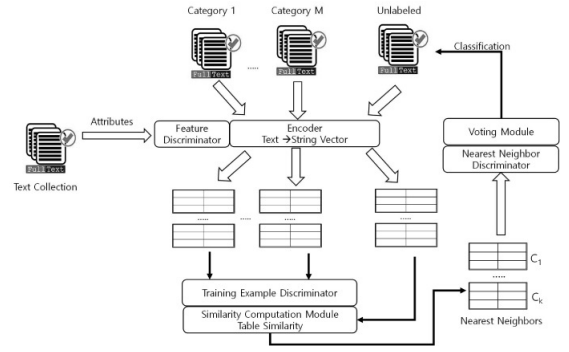


Figure 9. Text Categorization System

The connection of the index optimization with the text classification is illustrated in Figure 10. The texts each of which is labeled with its own domain are prepared as the sample texts, and the approach to the text classification is trained with them. A domain is assigned to a novice text by classifying it into a domain automatically. The role of the text classification system is to nominate the classifier which corresponds to the domain for the index optimization, automatically.

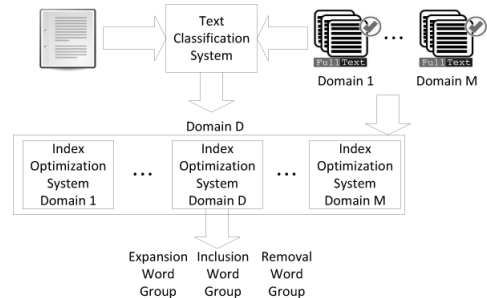


Figure 10. Index Optimization System connected with Text categorization

The process of executing the index optimization, connecting it with the text classification is illustrated in Figure 11. The sample texts each of which is labeled with its own domain are prepared, and the sample words each of which

is labeled with one of the three categories are prepared in each domain. A novice text is classified into one among the predefined domains, and the classifier which corresponds to the domain. The novice text is indexed into a list of words, and each word is classified into one of the three categories. The labeled words are the final output from this system; the domain which is assigned to the input text is the guide for selecting a classifier.

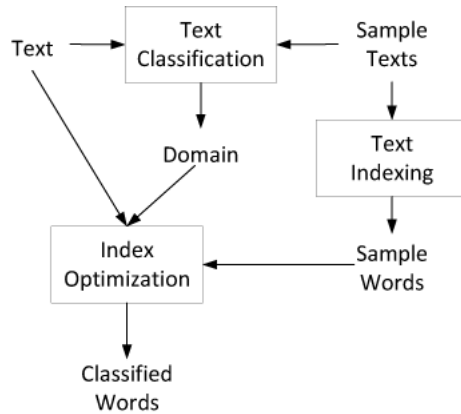


Figure 11. System Flow of Index Optimization connected with Text Categorization

Let us make some remarks on the connection of the index optimization with the text classification. The text classification is applied to the automatic decision of the input text domain. The role of text classification is to decide the domain for nominating a classifier, and the role of the index optimization is to classify words into expansion, inclusion, or removal. In the execution flow, the input text is classified into a domain, it is indexed into a list of words, and each word is classified into one of the three categories. We consider connecting the text classification as the main task the index optimization as the opposite to what is proposed in this research.

REFERENCES

- [1] T. Jo, "Table based K Nearest Neighbor for Word Categorization in News Articles", The Proceedings of 25th International Conference on Computational Science & Computational Intelligence, 2018.
- [2] T. Jo, "Optimizing Index of News Articles by Table based Version of K Nearest Neighbors", The Proceedings of 25th International Conference on Computational Science & Computational Intelligence, 2018.